[Continued on next page]

(54) Title: DETECTING PUBLIC NETWORK ATTACKS USING SIGNATURES AND FAST CONTENT ANALYSIS
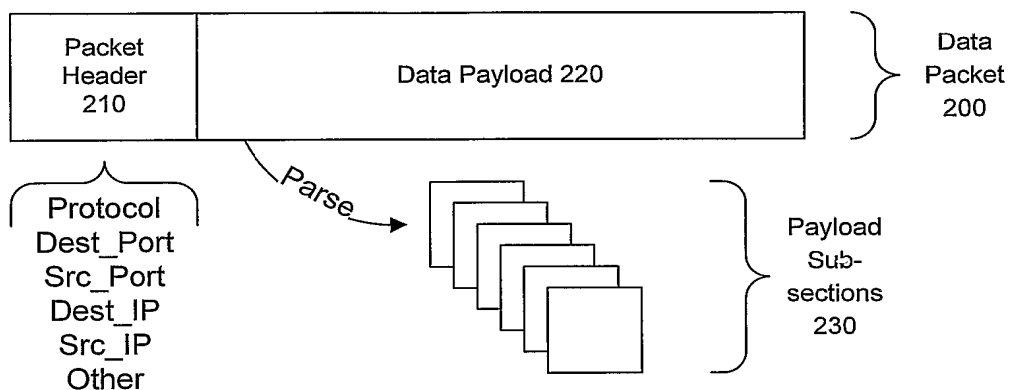
(57) Abstract: Network worms or viruses are a growing threat to the security of public and private networks and the individual computers that make up those networks. A content sifting method if provided that automatically generates a precise signature for a worm or virus that can then be used to significantly reduce the propagation of the worm elsewhere in the network or eradicate the worm altogether. The content sifting method is complemented by a value sampling method that increases the throughput of network traffic that can be monitored. Together, the methods track the number of times invariant strings appear in packets and the network address dispersion of those packets including invariant strings. When an invariant string reaches a particular threshold of appearances and address dispersion, the string is reported as a signature for suspected worm.

SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DETECTING PUBLIC NETWORK ATTACKS

USING SIGNATURES AND FAST CONTENT ANALYSIS

Background

1. Field of the Invention

[01] The present disclosure generally relates to the field of network security and more particularly relates to the prevention of self-propagating worms and viruses through data traffic analysis.

2. Related Art

[02] Many computers are connected to publicly-accessible networks such as the Internet. This connection has made it possible to launch large-scale attacks of various kinds against computers connected to the Internet. A large-scale attack is an attack that involves several sources and destinations, and which often (but not necessarily) involves a large traffic footprint. Examples of such large-scale attacks may include: (a) viruses, in which a specified program is caused to run on the computer, which then attempts to spread itself to other computers known to the host computer (e.g., those listed in the address book); and (b) denial of service attacks (DoS), in which a group of computers is exposed to so many requests that it effectively loses the ability to respond to legitimate requests. Many viruses and worms indirectly cause DoS attacks as well for networks by sending a huge amount of traffic while replicating. Distributed denial of service (DDOS) occurs when an attacker uses a group of machines (sometimes known as zombies) to launch a DoS attack.

[03] Another form of large-scale attack is called backdoor or vulnerability scanning. In such an attack an intruder scans for backdoors at machines or routers. A backdoor is a method by which a previously attacked machine can then be enlisted by future attackers to be part of future attacks.

[04]   Spam is unsolicited network messages often sent for commercial purposes. Large-scale spam is often simply the same as (or small variants of) the spam sent to multiple recipients. Note that this definition of spam includes both email as well as newer spam variants such as Spam Sent Over Instant Messenger.

[05]   A specific form of attack is an exploit, which is a technique for attacking a computer, which then causes the intruder to take control of the target computer, and run the intruder's code on the attack machine. A worm is a large-scale attack formed by an exploit along with propagation code. Worms can be highly efficacious, since they can allow the number of infected computers to increase geometrically. The worm can do some specific damage, or alternatively can simply take up network bandwidth and computation, or can harvest e-mail addresses or take any other desired action.

[06]   Many current worms propagate via random probing. In the context of the Internet, each of the number of different computers has an IP address, which is a 32-bit address. The probing can simply randomly probe different combinations of 32-bit addresses, looking for machines that are susceptible to the particular worm. Once the machine is infected, that machine starts running the worm code, and again begins probing the Internet. This geometrically progresses.

[07]   A very common exploit is a so-called buffer overflow. In computers, different areas of memory are used to store various pieces of information. One area in memory may be associated with storing information received from the network: such areas are often called buffers. However, an adjoining area in the memory may be associated with an entirely different function. For example, a document name used for accessing Internet content (e.g., a URL) may be stored into a URL buffer. However, this URL buffer may be directly adjacent to protected memory used for program access. In a buffer overflow exploit, the attacker sends a URL that is longer than the longest possible URL that can be stored in the receiver buffer and so overflows the URL buffer which allows the attacker to store the latter portion of its false URL into protected memory. By carefully crafting an extra long URL (or other message field), the attacker can overwrite the return address, and cause execution of specified code by pointing the return address to the newly

installed code. This causes the computer to transfer control to what is now the attacker code, which executes the attacker code.

[08] The above has described one specific exploit (and hence worm) exploiting the buffer overflow. A security patch that is intended for that exact exploit can counteract any worm of this type. However, the operating system code is so complicated that literally every time one security hole is plugged, another is noticed. Further, it often takes days for a patch to be sent by the vendor; worse, because many patches are unreliable and end users may be careless in not applying patches, it may be days, if not months, before a patch is applied. This allows a large window of vulnerability during which a large number of machines are susceptible to the corresponding exploit. Many worms have exploited this window of vulnerability.

[09] A signature is a string of bits in a packet that characterize a specific attack. For example, an attempt to execute the perl program at an attacked machine is often signaled by the string "perl.exe" in a message/packet sent by the attacker. Thus a signature-based blocker could remove such traffic by looking for the string "perl.exe" anywhere in the content of a message. The signature could, in general, include header patterns as well as exact bit strings, as well as bit patterns (often called regular expressions) which allow more general matches than exact matches.

[10] While the exact definition of the different terms above may be a matter of debate, the basic premise of these, and other attacks, is the sending of undesired information to a publicly accessible, computer, connected to a publicly accessible network, such as the internet.

[11] Different ways are known to handle such attacks. One such technique involves using the signature, and looking for that signature in Internet traffic to block anything that matches that signature. A limitation of this technique has come from the way that such signatures are found. The signature is often not known until the first attacks are underway, at which point it is often too late to effectively stop the initial (sometimes called zero-day) attacks.

[12]   An Intrusion Detection System (IDS) may analyze network traffic patterns to attempt to detect attacks. Typically, IDS systems focus on known attack signatures. Such intrusion detection systems, for example, may be very effective against so-called script kiddies who download known scripts and attempt to use them over again, at some later time.

[13]   Existing solutions to attacks each have their own limitations. Hand patching is when security patches from the operating system vendor are manually installed. This is often too slow (takes days to be distributed). It also requires large amounts of resources, e.g., the person who must install the patches.

[14]   A firewall may be positioned at the entrance to a network, and reviews the packets coming from the public portion of the network. Some firewalls only look at the packet headers; for example, a firewall can route e-mail that is directed to port 25 to a corporate e-mail gateway. The firewalls may be useful, but are less helpful against disguised packets, e.g., those disguised by being sent to other well-known services.

[15]   Intrusion detection and prevention systems, and signature based intrusion systems look for an intrusion in the network. These are often too slow (because of the time required for humans to generate a signature) to be of use in a rapidly spreading, new attack.

[16]   Other systems can look for other suspicious behavior, but may not have sufficient context to realize that certain behavior accompanying a new attack is actually suspicious. For example, a common technique is to look for scanning behavior but this is ineffective against worms and viruses that do not scan. This leads to so-called false negatives where more sophisticated attacks (increasingly common) are missed.

[17]   Scanning makes use of the realization that an enterprise network may be assigned a range of IP addresses, and may only use a relatively small portion of this range for the workstations and routers in the network. Any outside attempts to connect to stations within the unused range may be assumed to be suspicious. When multiple attempts are made to access stations within this address space, they may increase the level of

4

suspicion and make it more likely that a scan is taking place. This technique has been classically used as part of the so-called network telescope approach.

Summary

[18]   A content sifting system and method is provided that automatically generates a signature for a worm or virus. The signature can then be used to significantly reduce the propagation of the worm elsewhere in the network or eradicate the worm altogether. A complementary value sampling method and system is also provided that increases the throughput of network traffic that can be monitored. Together, the methods and systems identify invariant strings that appear in or across packets and track the number of times those invariant strings appear along with the network address dispersion of those packets that include the invariant strings. When an invariant string reaches a particular threshold of appearances and address dispersion, the string is reported as a signature for a suspected attack.

Brief Description of the Drawings

[19]   The details of the present invention, both as to its structure and operation, may be gleaned in part by study of the accompanying drawings, in which like reference numerals refer to like parts, and in which:

[20]   **Figure 1A** is a network diagram illustrating an example network according to an embodiment of the invention;

[21]   **Figure 1B** is a network diagram illustrating an example network according to an embodiment of the invention;

[22]   **Figures 2A – C** are block diagrams illustrating an example sensor unit according to an embodiment of the invention;

[23]   **Figure 3** is a block diagram illustrating an example packet according to an embodiment of the present invention;

[24]   **Figure 4** is a block diagram illustrating an example content prevalence table according to an embodiment of the invention;

[25]   **Figure 5** is a block diagram illustrating an example address dispersion table according to an embodiment of the invention;

[26]   **Figure 6** is a functional block diagram illustrating an example hashing technique according to an embodiment of the invention; and

[27]   **Figure 7** is a flow diagram illustrating an example process for identifying a worm signature according to an embodiment of the invention.

Detailed Description

[28]   The present description is related to United States patent application serial number **10/822,226** entitled DETECTING PUBLIC NETWORK ATTACKS USING SIGNATURES AND FAST CONTENT ANALYSIS, filed on April 8, 2004, which is incorporated herein by reference in its entirety.

[29]   Certain embodiments as disclosed herein provide for systems and methods for identifying an invariant string or repeated content to serve as a signature for a network attack such as a worm or virus. For example, one method and system as disclosed herein allows for a firewall or other sensor unit to examine packets and optimally filter those packets so that invariant strings within or across packets are identified and tracked. When the frequency of occurrence of a particular invariant string reaches a predetermined threshold and the number of unique source addresses and unique destination addresses also reach a predetermined threshold, the particular invariant string is reported as a signature for a suspected worm. For ease of description, the example embodiments described below refer to worms and viruses. However, the described systems and methods also apply to other network attacks and the invention is not limited to worms and viruses.

[30]   After reading this description it will become apparent to one skilled in the art how to implement the invention in various alternative embodiments and alternative applications. However, although various embodiments of the present invention will be described herein, it is understood that these embodiments are presented by way of example only, and not limitation. As such, this detailed description of various alternative

embodiments should not be construed to limit the scope or breadth of the present invention as set forth in the appended claims.

[31]   Fig. 1A is a network diagram illustrating an example network 30 according to an embodiment of the invention. In the illustrated embodiment, a computer 10, sensor unit 20, and an aggregator unit 40 are part of and are communicatively coupled via the network 30. The network 30 may be a local network, a wide area network, a private network, a public network, a wired network or wireless network, or any combination of the above, such as the ubiquitous Internet.

[32]   Internet messages are sent in packets including headers that identify the destination and/or function of the message. An IP header identifies both source and destination for the payload. A TCP header may also identify destination and source port number. The port number identifies the service which is requested from the TCP destination in one direction, and from the source in the reverse direction. For example, port 25 may be the port number used commonly for e-mail; port number 80 is often used for FTP and the like. The port number thus identifies the specific resources which are requested.

[33]   An intrusion is an attempt by an intruder to investigate or use resources within the network 30 based on messages over the network. A number of different systems are in place to detect and thwart such attacks. It has been recognized that commonalities between the different kinds of large-scale attacks, each of which attack a different security hole, but each of which have something in common.

[34]   Typical recent attacks have large numbers of attackers. Typical recent attacks often increase geometrically, but in any case the number of infected machines increases. Attacks may also be polymorphic, that is they change their content during each infection in order to thwart signature based methods.

[35]   The present systems and methods describe detecting patterns in data and using those patterns to determine the properties of a new attack. Effectively, this can detect an attack in the abstract, without actually knowing anything about the details of the attack. The detection of attack can be used to generate a signature, allowing automatic detection

of the attack. Another aspect describes certain current properties which are detected, to detect the attack.

[36]   A technique is disclosed which identifies characteristics of an abstract attack. This technique includes looking for properties in network data which make it likely that an attack of a new or previous type is underway.

[37]   The present disclosure describes a number of different properties being viewed; however it should be understood that these properties could be viewed in any order, and other properties could alternatively be viewed, and that the present disclosure only describes a number of embodiments of different ways of finding an attack under way.

[38]   An aspect of the disclosed technique involves looking through large amounts of data that is received by the sensor 20 as shown in Fig. 1A. One embodiment discloses a truly brute force method of looking through this data; and this brute force method could be usable if large amounts of resources such as memory and the like are available. Another embodiment describes scalable data reduction techniques, in which patterns in the data are determined with reduced resources, e.g., smaller configurations of memory and processing.

[39]   The computer 10 may be any of a variety of types of computing devices such as a general purpose computer device. The computer 10 may be a user device or a server machine or any other type of computer device that performs a multi-purpose or dedicated service.

[40]   The sensor 20 is configured with a data storage area 22. The sensor 20 may be any of a variety of types of computing devices such as a general purpose computer device. The sensor 20 may be a stand alone unit or it may be integral with the computer 10 or the aggregator 40.   There can be a single sensor 20 as shown or in other embodiments there can be a plurality of sensors that alone or collectively carry out the functions or a portion of the functions of the invention. The sensor 20 receives packets from the network 30 and analyzes the packets for indications of an attack. If a possible attack is detected, the sensor 20 can notify the aggregator 40, which can then take appropriate action.

[41]   Similarly, the aggregator 40 is configured with a data storage area 42 and may be any of a variety of types of computing devices such as the general purpose computer device. Additionally, there may be one or more aggregators 40 that alone or collectively carry out the functions or a portion of the functions of the invention.

[42]   Fig. 1B is a network diagram illustrating an alternative example network 60 according to an embodiment of the invention. In the illustrated embodiment, the computer 10 is communicatively coupled with an intrusion system 70 and a firewall 80 via the network 60. The computer 10 is also in communication with the Internet 90 via the firewall 80 or optionally through the intrusion system 70.

[43]   The intrusion system 70 is configured with a data storage area (not shown) and may be in communication with the firewall 80 via the network 60 or optionally through a direct communication link 75. The intrusion system 70 is also in communication with the Internet 90 through the firewall 80 or optionally directly through communication link 95. The intrusion system 70 preferably carries out the same function as the previously described sensor 20 and may be a stand alone unit or integrated with another device. In one embodiment, the intrusion system 70 can perform the combined functions of the previously described sensor 20 and the aggregator 40.

[44]   The intrusion system 70 may be any of a variety of types of computing devices such as a general purpose computer device. There may be a single intrusion system 70 as shown or there may be more than one that alone or collectively carry out the functions or a portion of the functions of the invention. In an embodiment, the intrusion system 70 may be integrated with the firewall 80 into a combined device 85. In such a case, the communication link 75 may take the form of shared memory or inter-process communication, as will be understood by one having skill in the art.

[45]   The firewall 80 is also configured with a data storage area (not shown) and may be any of a variety of types of computing devices such as a general purpose computer. Additionally, there may be one or more firewalls that alone or collectively carry out the functions or a portion of the functions of the invention described herein.

[46] Fig. 2A is a functional block diagram illustrating an example sensor 20 according to an embodiment of the invention. In the illustrated embodiment, the sensor 20 is configured with a data storage area 22 and includes a communication module 100, a destination checker module 110, a content analysis module 120, and a signature module 130. The data storage area 22 may include both internal and external data storage and include volatile and non-volatile memory devices. The configuration of computing devices with various types of memory is well known in the art and will therefore not be discussed in detail herein.

[47] The communication module 100 handles network communications for the sensor 20 and receives and processes packets appearing on the network interface (not shown). The communication module 100 may also handle communications with other sensors and one or more aggregators or computers. In one embodiment, when packets are received by the communication module 100, they are provided to the destination checker module 110, content analysis module 120, and signature module 130 for further processing in parallel.

[48] The destination checker module 110 examines packets based on a special assumption that there is known vulnerability in a destination machine. This makes the problem of detection much easier and faster. The destination checker module 110 analyzes the packets for known vulnerabilities such as buffer overflows at a specific destination port. For example, a list of destinations that are susceptible to known vulnerabilities is first consulted to check whether the destination of the current packet being analyzed is on the list. Such a list can be built by a scan of the network prior to the arrival of any packets containing an attack and/or can be maintained as part of routine network maintenance.) If the specific destination is susceptible to a known vulnerability, then the packet intended for that destination is parsed to determine if the packet data conforms to the vulnerability. For example, in a buffer overflow vulnerability for a URL, the URL field is found and its length is checked to see if the field is over a pre-specified limit. If the packet is determined to conform to a known vulnerability, delivery of that packet can be stopped. Alternatively, the contents of the packet that exploit the

vulnerability (for example, the contents of the field that would cause a buffer overflow) are forwarded as an anomalous signature, together with the destination and source of the packet. The contents may be forwarded, for example, to an aggregator 40 as previously described with respect to Fig. 1A so that a possible attack may be identified and stopped.

[49] Content analysis module 120 examines the content of a packet to determine if it meets criteria that are not necessarily based on a known vulnerability. For example, the content analysis module 120 may examine packets in the aggregate to determine if they contain repetitive content. It has been found that large attacks against network resources typically include content that repeats an unusual number of times. For example, the content could be TCP or IP control messages for denial of service attacks. By contrast, worms and viruses have content that contains the code that forms the basis of the attack, and hence that code is often repeated as the attack propagates from computer to computer. Spam has repeated content that contains the information the spammer wishes to send to a large number of recipients.

[50] Advantageously, only the frequently repeated content (signatures) are likely to be problems. For example, a signature that repeats just once could not represent a large-scale attack. At most, it represents an attack against a single machine. Therefore, the frequent signatures may be further analyzed by the content analysis module 120 to determine if it is truly a threat, or is merely part of a more benign message.

[51] The signature module 130 analyzes packet data to determine what signatures, if any, are included in the data payload. The signature module 130 may examine individual packets to find signatures or it may examine the data within a single packet and across packets to find signatures that extend across packet boundaries. The signature module 130 may work in concert with the other modules in the sensor 20 to provide them with information about signatures in packets.

[52] Fig. 2B is a functional block diagram illustrating an example content analysis module 120 according to an embodiment of the invention. In the illustrated embodiment, the content analysis module 120 is configured with a data storage area 22 and includes a

spreading module 122, a correlation module 124, an executable code detection module 126, and a scanning module 128.

[53]   The spreading module 122 is configured to determine whether a large (where "large" is defined by thresholds that can be set to any desired level) number of attackers or attacked machines are involved in sending/receiving the same content. The content is "common," in the sense that the same frequent signatures are being sent. During a large-scale attack, the number of sources or destinations associated with the content may grow geometrically. This is in particular true for worms and viruses. For spam, the number of destinations to which the spam content is sent may be relatively large; at least for large-scale spam. For denial of service attacks, the number of sources may be relatively large. Therefore, spreading content may be an additional factor representing an ongoing attack.

[54]   When a frequent signature is detected, the spreading module 122 investigates whether the content is exhibits characteristics of spreading. This can be done, for example, by looking for and counting the number of sources and destinations associated with the content.

[55]   In a brute force example, a table of all unique sources and all unique destinations is maintained. Each piece of content is investigated to determine its source and its destination. For each string S, a table of sources and a table of destinations are maintained. Each unique source or destination may increment respective counters. These counters maintain a count of the number of unique sources and unique destinations.

[56]   When the same string S comes from the same source, the counter is not incremented. When that same string does come from a new source, the new source is added as an additional source and the unique source counter is incremented. The destinations are counted in an analogous way. The source table is used to prevent over-counting the number of sources. That is, if Sally continually sends the message "hi Joe" Sally does not get counted twice.

[57]   The frequent and spreading signatures found by the spreading module 122 can also be subjected to additional checks such as a check for executable code, spam,

backdoors, scanning, and correlation. Each of these checks, and/or additional checks, can be carried out by modules, either software based, hardware based, or any combination thereof.

[58]   The correlation module 124 examines the source and destination of multiple packets to determine if an interval pattern is present. For example, a piece of content may be sent to a set of destinations in a first measured interval. In a later second measured interval, the same piece of content is sent by some fraction of these destinations acting as sources of the content. Such correlation can imply causality wherein infections sent to destinations in the first interval are followed by these stations acting as infecting agents in the later interval.

[59]   In one embodiment, a correlation test can be used to scalably detect the correlation between content sent to stations in one interval, and content sent by these sources in the next interval. The correlation test is a likely sign of an infection. Meeting the correlation test adds to the guilt score assigned to a piece of content.

[60]   For example, a bitmap for source addresses and a bitmap for destination addreses are initialized to "0" whenever a new signature is detected and added to what may be referred to as a frequent content table. A similar initialization occurs at the end of every time interval to reset the frequency. The concepts used are very similar to those described herein for detecting spreading content when similar bitmap structures can be used.

[61]   Thus, when a new signature is detected, the source IP address is hashed into the source bitmap and the destination IP address is analogously hashed into the destination bitmap. The bit positions set in the source bitmap for this interval are then compared with the bit positions set in the destination bitmap for the previous interval. If a large number of set bits are in common, it indicates that a large number of the destinations that received the content in the last interval are sending the same content in this interval. Accordingly, the correlation module 124 would identify that content as passing the correlation test.

[62]   Another example correlation test is a spam test conventionally known as the Bayesian spam test. The Bayesian test may heuristically analyze the content to determine if the suspected content is in fact spam according to the Bayesian rules.

[63]   The executable code detection module 126 detects the presence of executable code segments. The presence of executable code segments may also be an additional (but not necessary) sign of an attack. Worms and certain other attacks are often characterized by the presence of code (for example, code that can directly execute on Windows machines) in the attack packets they send. Therefore, in analyzing content to determine an infestation, the repeatable content is tested against parameters that determine executable code segments. It is unlikely that reasonably large segments of contiguous packet data will accidentally look like executable code; this observation is the basis of special techniques for determining the presence of code. In one aspect, a check is made for Intel 8086 and Unicode executable code formats.

[64]   In one embodiment, the executable code detection module 126 is configured to test each suspicious data segment that is identified. For example, a data segment starting at the beginning of a packet, at an offset, or spanning across packets can be tested for executable code. When code is detected to be over a specified length, the executable code detection module 126 reports a positive code test, for example to the sensor or intrusion system.

[65]   A variety of different code tests can be employed by the executable code detection module 126. For example, a particular code test can simply be a disassembler run on the packet at each of the plurality of offsets. Most worms and the like use 8086 code segments. Therefore, an 8086 disassembler can be used for this purpose.

[66]   Alternatively, a technique of looking for opcodes and associated information can be used as a code test. The opcodes may be quite dense, leaving only a few codes that are clearly not 8086 codes. Each opcode may have associated special information following the code itself. While a small amount of data may look like code, because of the denseness of the opcodes, it is quite unlikely that large strings of random data look like codes. For example, if 90% of the opcodes are assigned, a random byte of data has a

90% chance of being mistaken for a valid opcode; however, this is unlikely to keep happening when measured over 40 bytes of data that each of the appropriate bytes looks like a valid opcode.

[67]   This test, therefore, maintains a small table of all opcodes, and for each valid opcode the test uses the length of the instruction to test whether the bits are valid. In one example, the code test may start at offset O, peform a length test, and then repeat until a length greater than N for opcodes tests of length N. Then each bit at offset O along with its length in the opcode table, is looked up. If the opcode table indicates that the byte is invalid, the code test would fail.. If the opcode table entry is valid, the length test is incremented by the opcode table entry length and the code test would continue. The system thus checks for code at offset O by consulting the table looking for a first opcode at O. If the opcode is invalid, then the test fails, and the pointer moves to test the next offset. However, if the opcode is valid, then the test skips the number of bytes indicated by the instruction length, to find the next opcode, and the test repeats. If the test has not failed after reaching N bytes from the offset O, then the code test has succeeded.

[68]   This test can be carried out on each string, using 8086 and unicode, since most of the attacks have been written in these formats. It should be understood, however, that this may be extended to other code sets where desirable to do so.

[69]   As previously described, the code test can be combined with the frequent content test or other tests to confirm whether a piece of frequent content contains at least one fragment of code. In an alternative embodiment, the code detection test can be used as a threshold test prior to the other tests such as the frequent content test. In such an embodiment, only content that has a code segment of size N or more would be considered for frequent content testing.

[70]   The scanning module 128 is configured to determine whether IP addresses or ports are being probed for potential vulnerability. For example, it may be necessary for an attacker to communicate with vulnerable sources in order to launch an attack. Scanning may be used by the attacker or worm / virus to find valid IP addresses to probe for vulnerable services. Probing of unused addresses and/or ports can be used by the

attacker to make this determination. However it is possible that future attacks may also modify their propagation strategies to use pre-generated addresses instead of probing. Accordingly, one embodiment uses scanning only as an additional sign of an attack which is not necessary to output an anomalous signature.

[71] In one embodiment, a scanning test is employed that, unlike conventional scanning systems, uses both the content and the source as keys for the test. Conventional systems tested only the source address. In the scanning test, tests are made for content that is being sent to unused addresses (of sources that disburse such content and send to unused addresses) and not solely sources. A guilt score is assigned to pieces of "bad" content, though as a side-effect, the individual stations disbursing the bad content may also be tagged. Notice also that the exploit in a TCP-based worm will not be sent to these addresses because a connection cannot be initiated without an answer from the victim.

[72] In one embodiment, the scanning module 128 looks for a range of probes to an unused space. For example, a source address may make several attempts to communicate with an inactive address or port by mistake. A hundred attempts to a single unused address or port is less suspicious than a single attempt to each of a hundred unused addresses/ports. Thus rather than counting just the number of attempts to unused addresses, the scanning module 128 may also make an estimate of the range of unused addresses that have been probed.

[73] To implement these augmentations scalably, a representation of the set of the unused addresses/ports of an enterprise or campus network is maintained by the scanning module 128. For scalability, unused addresses can be done compactly using a bitmap (for example, for a Class B network, 64K bits suffices) or a Bloom Filter (described in Fan, et al., Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol, SIGCOMM 98, 1998). The list can be dynamically validated. Initial guesses about which address spaces are being used can be supplied by a manager. This can easily be dynamically corrected. For example, whenever an address S thought to be unassigned sends a packet from the inside, that address should be updated to be an assigned address. Note that in the special case of a contiguous address space, a simple network mask suffices.

[74]   A scalable list of unused ports can be kept by keeping an array with one counter for each port, where each array entry is a counter. The counter is incremented for every TCP SYN sent or each RESET sent, and decremented for every TCP FIN or FIN-ACK sent. Thus, if a TCP-based attack occurs to a port and many of the machines it contacts are not using this port, TCP FINs will not be sent back by these machines, or they will send TCP resets. Thus, the counter for that port will increase. Some care must be taken in implementing this technique to handle spoofing and asymmetrical routing, but even the simplest instance of this method will work well for most organizations.

[75]   A "blacklist" of sources that have sent packets to the unused addresses or ports in the last k measurement periods. This can be done compactly via a Bloom Filter or a bitmap. A hashed bit map can also be maintained, (similar to counting sources above) of the inactive destinations probed, and the ports for which scanning activity is indicated.

[76]   For each piece of frequent content, the mechanism keeps track of the range of sources in the blacklisted list associated with the content. Once again, this can be done scalably using a hashed bitmap as described herein. In one embodiment, testing for content of scanning can be implemented by hashing the source address of a suspicious signature S into a position within the bit map. When the number of bits set within that suspicion bit map exceeds a threshold, then the scanning is reported as true.

[78]   Note that while worms may evince themselves by the presence of reasonably large code fragments, other attacks such as Distributed Denial of Service may be based on other characteristics such as large amounts of repetition, large number of sources, and the reception of an unusually large number of TCP reset messages. The content analysis module 120 may identify spam, for example, as being characterized by repetitive presence of keywords identified based on heuristic criteria. These additional checks for spreading, correlation, executable code, scanning, and spam can be optional such that one or more or none of these tests may be used.

[77]   Fig. 2C is a functional block diagram illustrating an example signature module 130 according to an embodiment of the invention. In the illustrated embodiment, the signature module 130 is configured with a data storage area 22 and includes a parser

module 132, a filter module 134, a key module 136, and a data module 138. The signature module 130 is configured to examine packets for signatures that appear within a single packet or are spread across packets. The signature module 130 preferably works in connection with the other modules of the sensor 20 to detect a possible attack.

[78]   In an embodiment, the signature module 130 can perform a brute force examination of each packet that is received. It should be understood, however, that the brute force method of analyzing content could require incredible amounts of data storage. For example, commonly used intrusion systems / sensors that operate at 1 Gigabit per second, easily produce terabytes of packet content over a period of a few hours. Accordingly, a general data reduction technique may be used. It should be understood, however, that other detection techniques may be used without a general data reduction technique. Thus, in an embodiment, a data reduction technique can advantageously be used as part of those detection techniques that generate large amounts of data, such as signatures and source / destination addresses and ports.

[79]   In one aspect, a signature for a possible attack (also referred to as an "anomalous signature") may be established when any frequent content is found that also meets an additional test such as spreading, correlation, executable code segments, or any other test. According to another aspect, the signatures may be scored based on the amount of indicia they include. In any case, this information is used to form anomalous signatures that may then be used to block operations or may be sent to a bank of signature blockers and managers such as the aggregator 40 previously described with respect to Fig. 1A.

[80]   In addition to the signature, if a packet signature is deemed to be anomalous according to the tests above, the destination and source of the packet may be stored. This can be useful, for example, to track which machines in a network have been attacked, and which ones have been infected.

[81]   An intrusion detection system (or sensor) device may also (in addition to passing the signature, source, destination, or other information) take control actions by itself. Standard control actions that are well known in the state of the art include connection termination (where the TCP connection containing the suspicious signature is

terminated), connection rate limiting (where the TCP connection is not terminated but slowed down to reduce the speed of the attack), and packet dropping (where any packet containing the suspicious content is dropped with a certain probability). Note that when an attack is based on a known vulnerability, packet dropping with probability 1 can potentially completely prevent an attack from coming into a network or organization.

[82]   The signature module 130 is configured to identify a signature S from within any subset of the packet data payload and/or header. In general, a signature can be any subset of the data payload. A signature can also be formed from any portion of the data payload added to or appended to information from the packet header, for example, the TCP destination (or source) port. This type of signature recognizes that many attacks target specific destination (or in some limited cases, source) ports. An offset signature is based on the recognition that modern large-scale attacks may become polymorphic – that is, may modify the content on individual attack attempts. This is done to make each attack attempt look like a different piece of content. Complete content change is unlikely, however. Some viruses add small changes, while others encrypt the virus but add a decryption routine to the virus. Each contains some common piece of content; in the encryption example, the decryption routine would be the common piece of content.

[83]   The attack content may lie buried within the packet content and may be repeated, but other packet headers may change from attack to attack. Thus, according to another embodiment, the signature is formed by any continuous portion in the data payload, appended to the TCP destination port. Therefore, the signature module 130 investigates for content repetition strings anywhere within the TCP payload. For example, the text "hi Joe" may occur within packet 1 at offset 100 in a first message, and the same text "hi Joe" may occur in packet 2 at offset 200. This signature module 130 allows for counting these as two occurrences of the same string despite the different offsets in each instance.

[84]   The evaluation of this occurrence is carried out by evaluating all possible substrings in the packet of any certain length. A value of a substring length can be chosen, for example, 40 bytes. Then, a data payload each piece of data coming in may be

windowed, to first look for bytes 1 through 40, then look for bytes 2 through 41, then look for bytes 3 through 42. All possible offsets are evaluated.

[85]   Determining the length of substrings that are evaluated is a trade-off depending on the desired amount of processing. Longer substrings will typically have fewer false positives, since it is unlikely that randomly selected substrings can create repetitions of a larger size. On the other hand, shorter substrings may make it more difficult for an intruder to evade attacks.

[86]   Certain attacks may chop the attack into portions which are separated by random filler. However, these separated portions will still be found as several invariant content substrings within the same packet. In such an attack, a multi-signature may be identified by the signature module 130. A multi-signature may comprise one or more continuous portions of payload combined with information from the packet header such as the destination port.

[87]   Other attacks may break the attack into portions that are separated across two or more packets. In such an attack, when the packets are received and placed in order, the data payloads can be examined such that predetermined sized strings that span adjacent packets are analyzed for invariant content substrings that cross packet boundaries. Thus, an inter-packet signature may be identified that comprises a portion of payload from a first packet with a portion of payload from a second packet. Furthermore, the two source packets for the inter-packet signature are preferably adjacent when reordered.

[88]   The parser module 132 receives packets and parses the header and data payload from the packet. The parser module 1320 additionally extracts information from the packet header such as the protocol, the source IP address, the destination IP address, the source IP (or UDP) port, and the destination IP (or UDP) port just to name a few. The parser module 132 also breaks down the data payload into predetermined sized strings for further processing by other components of the sensor. As described, the predetermined sized strings may extend across packet boundaries such that a single predetermined sized string may have a portion of its content from a first packet and a portion of its content from a second, adjacent packet.

[89]   The filter module 134 may be implemented in hardware as a series of parallel processors or application specific integrated circuits. Alternatively the filter module 134 may be implemented in software that includes one or more routines. Advantageously, the software may be threaded so that the filtering process implemented in software is also a parallel process to the extent allowed by the associated hardware on which the software is running.   The function of the filter 134 is to optimally reduce the number of predetermined sized strings that are processed while maintaining high efficacy for virus detection, as described later in detail with respect to Fig. 6.

[90]   The key manager 136 identifies the invariant strings from the data payload that may qualify as a signature, for example, due to their repetitive nature, inclusion of code segments, matching a predetermined string, etc. The key manager 136 may combine information from the packet header with an identified string of content from the packet data payload to create a key. Each key is possibly a worm or virus signature. Alternatively, the key manager 136 may create a key from the string of content alone or from the string of content in combination with other information selected from the packet header such as the destination IP address or the destination port. In an embodiment, the key manager 136 performs data reduction on the key to minimize the size of the key.

[91]   In one embodiment, a data reduction technique called hashing may be employed. Hashing is a set of techniques to convert a long string or number into a smaller number. A simple hashing technique is often to simply remove all but the last three digits of a large number. Since the last three digits of the number are effectively random, it is an easy way to characterize something that is referred by a long number. For example, U.S. Patent No. 6,398,311 can be described simply the 311 patent. However, much more complex and sophisticated forms of hashing are known.

[92]   In one example, assume the number 158711, and that this number must be assigned to one of 10 different hashed "bins" by hashing the number to one of 10 bins. One hashing technique simply adds the digits 1 + 5 + 8 + 7 + 1 + 1 equals 23. The number 23 is still bigger than the desired number of 10. Therefore, another reduction

technique is carried out by dividing the final number by 10, and taking the remainder ("modulo 10"). The remainder of 23 divided by 10 is 3. Therefore, in 158711 is assigned to bin 3. In this technique, the specific hash function is: (1) add all the digits; and (2) take the remainder when divided by 10.

[93] The same hash function can be used to convert any string into a number between 0 and 9. Different numbers can be used to find different hashes.

[94] The hash function is repeatable, that is, any time the hash function receives the number 158711, it will always hash to bin 3. However, other numbers will also hash to bin 3. Any undesired string in the same bin as a desired string is called a hash collision.

[95] Many other hash functions are known, and can be used. These include Cyclic Redundancy Checks (CRCs) commonly used for detecting errors in packet data in networks, a hash function based on computing multiples of the data after division by a pre-specified modulus, the so-called Carter-Wegman universal hash functions (the simplest instantiation of which is to multiply the bit string by a suitably chosen matrix of bits), hash functions such as Rabin hash functions based on polynomial evaluation, and one-way hash functions such as MD-5 used in security. This list is not exhaustive and it will be understood that other hash functions and other data reduction techniques can be used.

[96] A data reduction technique that is advantageous to use with the data payload subsections 230 described with respect to Fig. 3 allows adding a part of the hash and removing a part when moving between two adjacent subsections. One aspect of this embodiment, therefore, may use an incremental hash function. Incremental hash functions make it easy to compute the hash of the next substring based on the hash of the previous substring. One classic incremental hash function is a Rabin hash function (used previously by Manber in spotting similarities in files instead of other non-incremental hashes (e.g, SHA, MD5, CRC32)).

[97] A large data payload may contain thousands of bytes. Accordingly, to minimize the probability of hash collisions (where different source strings result in the same value after hashing) the data reduction may be, for example, a hash to 64 bits.

[98] The string S that is hashed may also include information about the destination port. The destination port generally remains the same for a worm, and may distinguish frequent email content from frequent Web content or peer-to-peer traffic in which the destination port changes.

[99] In an embodiment, use of the Rabin hash function (also called the Rabin fingerprint) advantageously simplifies the analysis of data across packets. In an embodiment, the last 40 byte subsection of the data payload of a packet is stored after the packet processing is complete. The Rabin fingerprint for that subsection is also stored. When the next data payload is analyzed, the Rabin fingerprint is computed for the 40 byte subsection that includes the last 39 bytes of the previous packet and the first byte from the new packet. In this fashion, the packets may be examined and analyzed as a continuous stream of data – across packet boundaries. This allows the detection of an attack that spreads invariant strings across packets.

[100] After a signature or key is created, the data manager 138 processes the signature. In an embodiment, the signature is subjected to a frequent signature test. Each key can be stored in a database. For example, the data manager 138 may maintain a content prevalence table and an address dispersion table (described later with respect to Figs. 4 and 5, respectively). The content prevalence table includes entries for keys and the number of times the particular key has been encountered ("count"). If a newly generated key is not present in the address dispersion table, the key is placed in the content prevalence table for tracking of the number of times the key is encountered. When the count for a particular key in the content prevalence table exceeds a predetermined threshold, the data manager 138 moves the key into the address dispersion table. In an embodiment, the content prevalence and address dispersion tables may be periodically flushed or specific entries may individually expire after a predetermined time period.

[101] Fig. 3 is a block diagram illustrating an example packet 200 according to an embodiment of the present invention. In the illustrated embodiment, the packet 200 comprises a header 210 and a data payload 220. The header 210 typically includes information relevant to the packet 200 such as the protocol by which the packet should be

processed, the source IP address, the source IP port, the destination IP address, and the destination IP port. Other information may also be advantageously located in the header 210.

[102] The data payload 220 can be very large and is preferably divided up into smaller more manageable sized chunks, for example by the aforementioned parser. These more manageable sized chunks are shown as payload subsections 230. The size of a payload subsection can vary and is preferably optimized based on the processing power of the sensor 20, available memory 22, and other performance or result oriented parameters. In one embodiment, the size of a payload subsection 230 is 40 bytes.

[103] Alternatively, the data payload subsections can be all of the contiguous strings in the data payload of any length. Or the subsections may be all of the contiguous strings in the data payload with the same length. Other possible combinations of data payload subsections may also be employed as will be understood by those skilled in the art. In a preferred embodiment, each subsection is 40 bytes, with the first subsection comprising bytes 1 – 40; the second subsection comprising bytes 2 – 41; the third subsection comprising bytes 3 – 42; and so on until each byte in the data payload is included in at least one subsection.

[104] Fig. 4 is a block diagram illustrating an example content prevalence table 250 according to an embodiment of the invention. In the illustrated embodiment, each row of the content prevalence table 250 includes a key and a count. For example, the count may represent the number of times the specific key has been encountered. As previously described, the key may be a string from the data payload of a packet and may also include the protocol and / or destination port information from the packet header. Alternatively, the key may be a representation of the string from the data payload (or the string combined with header information) after a data reduction has been performed.

[105] In an embodiment, the data manager 138 (previously described with respect to Fig. 2C) may maintain the content prevalence table 250. For example, when an new key is identified, the key is looked up in the content prevalence table 250. If the key is not in

the table, it is added to the table along with a count of 0. Alternatively, if the key is already in the table, then the count associated with the key is incremented.

[106] Additionally, a frequency threshold can also defined. Thus, if the count for a particular key exceeds the frequency threshold, then the key is identified as a frequent or repetitive key. In an alternative embodiment, a time threshold may also be defined for each entry in the content prevalence table 250. Accordingly, when the time threshold is reached for a particular entry, the counter can be reset so that the frequent content test effectively requires the key to be identified a certain number of times during a specified time period.

[107] Fig. 5 is a block diagram illustrating an example address dispersion table 270 according to an embodiment of the invention. In the illustrated embodiment, each row of the address dispersion table 270 includes a key and a count of the unique source IP addresses and a count of the unique destination IP addresses associated with the key. When a particular key in the content prevalence table 250 is identified as being a frequent or repetitive key, the data manager preferably creates an entry in the address dispersion table 270 for that key. Alternatively, when the key manager identifies a key that already exists in the address dispersion table 270, the relative counts for the unique source IP address and the unique destination IP address is updated if necessary.

[108] Because the tables illustrated in Figs. 4 and 5 may become quite huge in practice, data reduction techniques may be performed to manage the content prevalence and the address dispersion tables. For example, a data reduction hash may be performed on one or both of the tables 250 and 270.

[109] In an embodiment, an optional front end test such as a Bloom Filter (described in Burton Bloom: Space/Time Tradeoffs In Hash Coding With Allowable Errors; Communications ACM, 1970) or a counting Bloom Filter (described in Fan, et al., Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol, SIGCOMM 98, 1998) to sieve out content that is repeated only a small number of times.

[110] Fig. 6 is a functional block diagram illustrating an example hashing technique according to an embodiment of the invention. In general, for a more scalable storage of

content in the content prevalence and / or address dispersion tables, a certain number k of hash stages are established. Each stage I hashes the value S using a specified hash function $Hash(I)$, where $Hash(I)$ is a different hash function for each stage I. For each of those stages, a specific position, $k(I)$ is obtained from the hashing. The counter in position $k(I)$ is incremented in each of the k stages. Then, the next I is established. Again, there are k stages, where k is often at least three, but could even be 1 or 2 in some instances.

[111] The data reduction hashing system checks to see if all of the k stage counters that are incremented by the hash for a specific string S are greater than a stage frequency threshold. S is added to the frequent content table only when all of the k counters are all greater than the threshold.

[112] Specifically with respect to Fig. 6, when k=3, the data reduction technique would be called a 3 stage hash. Each stage is a table of counters which is indexed by the associated hash function (Hash(I)) that is computed based on the packet content. At the beginning of each measurement interval, all counters in each stage are initialized to 0. Each packet comes in (e.g., Packet S) and is hashed by a hash function associated with the stage. The result of the hash is used to set a counter in that stage.

[113] For example, the packet S is hashed by a stage 1 hash function. This produces a result of 2, shown incrementing counter 2 in the stage 1 counter array. The same packet S is also hashed using the stage 2 hash function, which results in an increment to counter 0 in the stage 2 counter array. Similarly, the packet S is hashed by the stage 3 hash function, which increments counter 6 of the stage 3 counter array. In this example, the same packet hashes to three different sections (in general, though there is a small probability that these sections may coincide) in the three different counter stages.

[114] After the hashing, the stage detector 290 determines if the counters that have currently been incremented are each above the frequency threshold. The signature is added to the frequent content memory 295 only when all of the stages have been incremented above the stage frequency threshold.

26

[115] As examples, the stage 1 hash function could sum digits and take the remainder when divided by 13. The stage 2 hash function could sum digits and take the remainder when divided by 37 and the stage 3 hash function could be a third independent function. In an embodiment, parameterized hash functions may be used, with different parameters for the different stages, to produce different but independent instances of the hash function.

[116] The use of multiple hash stages with independent hash functions reduces the problems caused by multiple hash collisions. Moreover, the system is entirely scalable. By simply adding another stage, the effect of hash collisions is geometrically reduced. Moreover, since the memory accesses can be performed in parallel, this can form a very efficient, multithreaded software or hardware implementation.

[117] Advantageously, the bits in the individual stage counter arrays can be weighted by the probability of hash collisions, in order to get a more accurate count. When counting source and destination IP address, the weighting provides a more accurate count of the number of unique sources and destinations. Additionally, when applied to counting IP addresses, this technique effectively creates and stores a bitmap, where each bit represents an IP address. Advantageously, the storage requirements are significantly reduced, rather than storing the entire 32-bit IP address in an address table.

[118] While the bitmap solution is better than storing complete addresses, it still may require keeping hundreds of thousands of bits per frequent content. Another solution carries out even further data compression by using a threshold T which defines a large value. For example, defining T as 100, this system only detects values that are large in terms of source addresses. Therefore, no table entries are necessary until more than 100 source addresses are found.

[119] It also may be desirable to know not only the number of source addresses, but also the rate of increase of the source addresses. For example, it may be desirable to know that even though a trigger after 100 sources is made, that in the next second there are 200 sources, in the second after that there are 400 sources, and the like.

[120]   In an embodiment, even more scaling is achievable to advantageously use only a small portion of the entire bit map space. For example, if an identified signature is a frequent signature the IP address is hashed to a W bit number S_{HASH}. Only certain bits of that hash are selected, e.g. the low order r bits. That is, this system scales down the count to only sampling a small portion of the bitmap space. However, the same scaling is used to estimate the complete bitmap space. The same scaling down operations are also carried out on the destination address.

[121]   For example, an array of 32-bits (i.e., r = 32) may be maintained, where the threshold T is 96. Each source address of the content is hashed to a position between 1 and 96. If the position is between 1 and 32, then it is set. If the position is beyond 32, then it is ignored, since there is no portion in the array for that bit.

[122]   At the end of a time interval, the number of bits set into the 32-bit array is counted, and corrected for collisions. The value is scaled up based on the number of bits which were ignored. Thus, for any value of T, the number of bits set within the available portion of the registers is counted, and scaled by a factor of T. For example, in the previous example, if we had hashed from 1 to 96 but only stored 1 through 32, the final estimate would be scaled up by a factor of 3.

[123]   This technique may also be used to count a rising infection over several intervals, by changing the scaling factor. For example, a different scaling factor is stored along with the array in each interval. This technique can, therefore, reliably count from a very small to a very large number of source addresses with only a very small number of bits, and can also track rising infection levels.

[124]   Accordingly, the address is hashed and a scale factor for source addresses is assigned to a variable, e.g., SourceScale. If the high order bits of the hash from positions r+1 to r + SourceScale are all zero, the low order r bits are used to set the corresponding position in the source bit map. For example, if SourceScale is initially 3 and r is 32, essentially all but the low order 35 bits of the hash are ignored and the low order 32 bits of the 35 bits are focused on, a scaling of $2^{(35-32)} = 2^3 = 8$.

[125] When the time interval ends, the counter is cleared, and the variable (SourceScale) is incremented by some amount. If, in the next interval the scale factor goes up to 4, the scaling focuses on the top 36 bits of the hash, giving a scaling of $2^4 = 16$. Thus by incrementing the variable (SourceScale) by 1, the amount of source addresses that can be counted is doubled. Thus when comparing against the threshold for source addresses, the number of bits in the hash is scaled by a factor of $2^{(SourceScale - 1)}$ before being compared to the threshold. This same technique can also be used for destination IP addresses.

[126] Fig. 7 is a flow diagram illustrating an example process for identifying a worm signature according to an embodiment of the invention. At a high level, a network attack can be detected by receiving a plurality of packets on a network and analyzing the data payloads of those packets to detect common content among the packets. Data reduction techniques may also be employed to optimize the high level process. For example, initially, in step 300, the sensor receives a packet. For example, the previously described communication module may receive the packet. Upon receipt, the packet is parsed in step 310 and header information is extracted and the data payload is divided up into a plurality of strings .(subsections) as shown in step 320. In an embodiment, the parsing function may be carried out by the previously described parser module. In a brute force method, the data payload may be divided up into the universe of all possible strings of one or more characters that are present in the data payload. Such an operation, however, is computationally expensive.

[127] Alternatively, the data payload may be divided up into the universe of all strings having a minimum length. While this further reduces the number of strings relative to the minimum length, the operation remains computationally expensive.

[128] In an embodiment, the data payload may be divided up into the universe of all strings of a specific length. This operation significantly reduces the number of strings created by the parsing of the data payload without compromise because each string created is representative of all longer strings including the baseline string.

Advantageously, the specific string length can be optimized for detecting invariant strings in viruses and worms.

[129]   Additionally, if a specific length subsection is employed, then a portion of the data from the previous packet and a portion of the data from the current packet can be combined to create specific length subsections that span the packet boundary, as illustrated in step 330. For example, if the specific length was 40 bytes, then the last 39 bytes of the data payload of the previous packet and the first byte from the data payload of the current packet can be combined to create a single subsection.

[130]   Once the data payload has been parsed into subsections, and combined with portions from an adjacent packet, the subsections are then filtered, as shown in step 340. In one embodiment, the filtering function may be carried out by the previously described filter module.   The filtering may be carried out in a series of multi-stage hardware components or it may be carried out in software. The function of the filtering is to reduce the number of data payload subsections that require processing. In an embodiment, the Rabin fingerprint is calculated for each subsection and then only those subsections meeting a predetermined criteria are processed further. For example, after the Rabin fingerprint is calculated, each subsection that ends with six (6) zeroes is processed further. This may have the effect of thinning the number of subsections requiring processing to a fraction of the original number, for example to $1/64^{th}$ of the original number. Furthermore, because Rabin fingerprinting is randomly distributed, the creator of a worm or virus cannot know which subsections will be selected for further processing.

[131]   In one example, if the specific string length is 40 bytes and the thinning ratio is 1/64, the probability of tracking a worm with a signature of 100 bytes is 55%. However, the probability of tracking a worm with a signature of 200 bytes is 92% and the probability of tracking a worm with a signature of 400 bytes is 99.64%. Notably, all known worms today have had invariant content (i.e., a signature) of at least 400 bytes.

[132]   After the data payload subsections have been filtered, processing step 350 can be undertaken to determine if all of the subsections have been processed. If they have, then

the process loops back to receive the next packet in step 330. If all of the subsections have not been processed, then in step 360 a key is created for each data subsection. In one embodiment, the key creation may be carried out by the previously described key module. The key preferably includes the protocol identifier, the destination port, and the Rabin fingerprint for the data subsection. Alternatively, the key can include the Rabin fingerprint alone, or the Rabin fingerprint and the protocol, or the Rabin fingerprint and any combination of other identifying information. Once the key is created, the address dispersion table is consulted in step 370 to see if the key exists in the table. If the key does not exist in the address dispersion table, then the content prevalence table is updated in step 380 accordingly and the count for the key is initialized if the entry is new or incremented if the key was already present in the content prevalence table. If the count is incremented and the new count exceeds the predetermined threshold number as determined in step 390, then an entry for the key is created in the address dispersion table as illustrated in step 400. If the count does not exceed the threshold (or after the address dispersion table entry is created), the process returns to step 350 to determine if all subsections have been processed.

[133] After an entry is created in the address dispersion table in step 400, then the entry is updated to fill in the necessary information. Additionally, back in step 340, if it is determined that an address dispersion table entry exists for the particular key, then the entry is updated. For example, the count for the source IP address and the count for the destination IP address can be updated in step 410 if those addresses are unique and have not yet been associated with the particular key.

[134] After the address counts are updated, it is determined in step 420 if the new counts exceed a predetermined threshold. If they do, then in step 430 the key is reported (e.g., to the aggregator) as a possible signature for a suspected worm. Additionally, the packet (or packets if the data for the key came from two adjacent packets) containing the key are also reported. If the counts do not exceed the threshold (or after the report has been made), then the process returns to the step 350 to determine if all subsections have been processed.

[135]   Those of skill will further appreciate that the various, illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein can often be implemented as electronic hardware, computer software, or combinations of both.   To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality.   Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system.   Skilled persons can implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the invention.   In addition, the grouping of functions within a module, block, circuit or step is for ease of description.   Specific functions or steps can be moved from one module, block or circuit without departing from the invention.

[136]   The various illustrative logical blocks, modules, and circuits described in connection with the embodiments disclosed herein can be implemented or performed with a general purpose processor, a digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, or any combination thereof designed to perform the functions described herein.   A general-purpose processor can be a microprocessor, but in the alternative, the processor can be any processor, controller, microcontroller, or state machine.   A processor can also be implemented as a combination of computing devices, for example, a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

[137]   The steps of a method or technique described in connection with the embodiments disclosed herein can be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two.   A software module can reside in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium.   An

exemplary storage medium can be coupled to the processor such the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium can be integral to the processor. The processor and the storage medium can reside in an ASIC.

[138] The above description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the invention. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein can be applied to other embodiments without departing from the spirit or scope of the invention. Thus, the invention is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

WHAT IS CLAIMED IS:

1.   A computer implemented method for analyzing network activity, comprising:

obtaining a portion of data to be analyzed to determine a network attack;

carrying out a data reduction on said portion to reduce said data portion to a reduced data portion in a repeatable manner; and

analyzing a plurality of said reduced data portions to detect common elements within said reduced data portion, said analyzing reviewing for common content indicative of a network attack.

2.   A method as in claim 1, wherein said analyzing common content comprises determining frequently occurring sections of message information within said reduced data portion.

3.   A method as in claim 1, wherein said analyzing common content comprises determining increasing number of sources that are sending within said portions; and determining increasing number of destinations that are receiving within said portions.

4.   A method as in claim 1, further comprising analyzing for the presence of a specified type of code within said data.

5.   A method as in claim 2, further comprising after said analyzing determines said frequently occurring sections of message information, carrying out an additional test on said frequently occurring sections of message information.

6.   A method as in claim 5, wherein said additional test comprises identifying an increasing number of at least one of sources and destinations of said frequently occurring sections of message information.

7.   A method as in claim 5, wherein said additional test includes identifying code within the frequently occurring sections.

8.   A method as in claim 1, wherein said data reduction includes carrying out a hash function on said portion of data.

9.   A method as in claim 2, wherein said determining frequently occurring sections further comprises using at least first, second and third data reduction techniques on each said portion; obtaining at least first, second and third results; counting said first, second and third results; and establishing frequently occurring sections when all of said at least first second and third results have a frequency of occurrence greater than a specified amount.

10.   A method as in claim 1, wherein said portion of data at least includes a portion of the data payload.

11.   A method as in claim 5, wherein said additional test comprises maintaining a first list of unassigned addresses; forming a second list of sources that have sent to addresses on said first list; and comparing a current source of a frequently occurring section to said second list.

12.   A method as in claim 11, further comprising data reducing information in said first list and said second list.

13.   A method as in claim 5, wherein said additional test comprises:
first monitoring a first content sent to a destination;
second monitoring a second content sent by said destination; and
determining whether there is a correlation between said first content and said second content.

14.   A method as in claim 13, wherein said first monitoring comprises monitoring multiple destinations, and said second monitoring comprises monitoring multiple destinations during a different time period than said first monitoring.

15.   A method as in claim 14, wherein said first and second monitoring comprises data reducing information about said destinations, and storing at least one table about said data reduced information.

16.   A method as in claim 10, wherein said portion of data further includes a portion of a packet header.

17.   A method as in claim 16, wherein said portion of a packet header is a port number indicating a requested service.

18.   A method as in claim 16, wherein said portion of a packet header is a port number indicating a destination port.

19.   A method as in claim 1, further comprising forming a plurality of portions from each packet, each of said plurality of portions comprising a subset of the packet.

20.   A method as in claim 19, wherein each of said plurality of portions comprises a continuous portion from the packet, and information indicative of a port number indicating a service requested by a network packet.

21.   A method as in claim 2, wherein said determining frequently occurring sections comprises:
       taking a first hash function of said portion,
       maintaining a first counter, with a plurality of stages, and incrementing one of said stages based on said first hash function;
       taking a second hash function of said portion; and
       maintaining a second counter, with a plurality of stages, and incrementing one of said stages of said second counter based on said second hash function.

22.   A method as in claim 21, further comprising checking said one of said stages of said first counter and said one of said stages of said second counter against a threshold, and identifying said portion as frequent content only when both said one of said stages of said first counter and said one of said stages of said second counter are each above said threshold.

23.   A method as in claim 22, further comprising adding frequent content to a frequent content table.

24.   A method as in claim 8, wherein said hash function is an incremental hash function.

25.   A method as in claim 2, further comprising monitoring for scanning of addresses associated with said frequent content.

26.   A method as in claim 7, wherein said detecting code comprises looking for a first valid opcode at a first location, based on said first valid opcode, determining a second location representing an offset of said first valid opcode, and looking for a second valid opcode at said second location.

27.   A method as in claim 26, further comprising establishing the reduced data portion as including code when a predetermined number of valid opcodes are found at proper distances.

28.   A method as in claim 1, further comprising, determining a list of first computers that are susceptible to a specified attack, and monitoring only messages directed to said first computers for said specified attack.

29.   The method of claim 28 where said monitoring comprises checking for a message that attempts to exploit a known vulnerability to which a computer is vulnerable, as said specified attack.

30.   A method as in claim 29, wherein said checking comprises checking for a field that is longer than a specified length.

31.   The method of claim 1, wherein said portion of data is obtained from at least two discrete packets.

32.   The method of claim 1, wherein each reduced data portion in said plurality of reduced data portions is identified by a common characteristic in the reduced data portion.

33.   A system for detecting a network attack, comprising:
a communication module configured to receive a plurality of packets on a

network;

a signature module configured to receive said plurality of packets from the communication module and analyze the content of said packets to detect common content among said packets to identify a network attack.

34.   The system of claim 33, wherein the signature module further comprises:

a parser module configured to parse a packet into a plurality of portions;

a key module configured to perform a data reduction on each of said plurality of portions to create a plurality of data reduced portions; and

a data module configured to store said data reduced portions in a data storage area.

35.   The system of claim 34, further comprising a filter module configured to reduce the number of said plurality of data reduced portions prior to storage.

36.   The system of claim 33, further comprising a content analysis module configured to analyze the common content of said plurality of packets to identify a network attack.

37.   The system of claim 36, wherein the content analysis module further comprises a spreading module configured to determine whether a network attack is spreading to additional network devices.

38.   The system of claim 36, wherein the content analysis module further comprises a correlation module configured to determine whether packets sent in a first interval to a destination address are sent from said destination address in a second interval.

39.   The system of claim 36, wherein the content analysis module further comprises a code detection module configured to identify the presence of executable code in a packet.

40.   The system of claim 36, wherein the content analysis module further comprises a scanning module configured to determine if a network device is being probed for network vulnerabilities.

41.   The system of claim 33, further comprising a destination checker module configured to identify packets addressed to an address with a known network vulnerability.

42.   The system of claim 41, wherein the destination checker module is further configured to analyze the content of packets addressed to a an address with a known network vulnerability.

43.   A computer implemented method for analyzing network activity, comprising:

receiving a plurality of packets transiting a network;

analyzing the content of said plurality of packets to detect common content among said packets; and

identifying network attacks based upon said analysis for common content.

44.   The method of claim 43 wherein analyzing the content further comprises analyzing source information in the plurality of packets.

45.   The method of claim 43 wherein analyzing the content further comprises analyzing destination information in the plurality of packets.

46.   The method of claim 43 wherein analyzing the content further comprises performing a data reduction on at least a portion of each of the packets of said plurality of packets to form a plurality of data reduced packets.

47.   The method of claim 43 further comprising comparing the destinations of said plurality of packets with destinations having known vulnerabilities.

48.   The method of claim 47 further comprising analyzing the content of packets with destinations having known vulnerabilities.

49.   The method of claim 43 wherein identifying further comprises identifying one or more signatures of an attack.

50.   The method of claim 43, wherein analyzing the content includes analyzing the content of the payloads in the plurality of packets.

51.   The method of claim 46 wherein said data reduction comprises performing a hash function on at least a portion of each of the packets.

52.   The method of claim 51 wherein said hash function is an incremental hash function.

53.   The method of claim 43 further comprising determining whether there is an increasing number of sources and destinations of packets having common content.

54.   The method of claim 43 further comprising analyzing the content of said plurality of packets for the presence of a specified type of code.

55.   The method of claim 43 further comprising forming a plurality of portions from each of said plurality of packets, each portion comprising a specified subset of a packet.

56.   The method of claim 46, wherein a single portion comprises data from at least two packets.

57.   The method of claim 46, wherein analyzing further comprises analyzing a subset of the data reduced portions, the subset identified by a common characteristic in each of the data reduced portions.

58.   A computer implemented method for analyzing network activity comprising:

   obtaining a plurality of packets being transmitted across a network;

   performing a data reduction on at least a portion of each of the packets of said plurality of packets to form a plurality of data reduced packets;

   detecting repetition of content among said plurality of data packets;

analyzing sources and destinations of said plurality of packets; and

identifying network attacks based upon said detection of repetition and said analysis of sources and destinations.

59.   The method of claim 58 wherein the sources and destinations of said plurality of packets are only analyzed for packets wherein a repetition of content has been detected.

60.   The method of claim 58 further comprising comparing the destinations of said plurality of packets with destinations having known vulnerabilities.

61.   The method of claim 60 further comprising analyzing the content of packets with destinations having known vulnerabilities.

62.   The method of claim 58 wherein identifying further comprising identifying one or more signatures of an attack.

63.   The method of claim 60, wherein analyzing the content includes analyzing the content of the payloads in the plurality of packets.

64.   The method of claim 63 further comprising determining whether there is an increasing number of sources and destinations of packets having common content.

65.   The method of claim 60 further comprising analyzing the content of said plurality of packets for the presence of a specified type of code.

66.   The method of claim 58 wherein said data reduction comprises carrying out a hash function on at least a portion of each of the packets.

67.   The method of claim 58 wherein said detecting common content comprises using at least first, second and third data reduction techniques on at least a portion of each of the packets, to obtain at least first, second and third results, and to count said first, second and third results, and to detect repetition when all of said at least first second and third results have a frequency of occurrence greater than a specified amount.

68.   The method of claim 65 further comprising maintaining a first list of addresses; forming a second list of sources that have sent to addresses on said first list; and comparing a current source of common content to said second list.

69.   The method of claim 58 further comprising,

monitoring a first content sent to a destination;

monitoring a second content sent by said destination; and

determining a whether there is a correlation between said first content and said second content.

70.   The method of claim 58 further comprising forming a plurality of portions from each of said plurality of packets, each portion comprising a specified subset of a packet.

71.   The method of claim 70, wherein a single portion comprises data from at least two packets.

72.   The method of claim 70, wherein forming a plurality of portions further comprises identifying all portions comprising a specified subset of a packet; and forming a plurality of portions from a subset of said all portions, wherein each portion in the subset has a common characteristic.

73.   The method of claim 72, wherein the specified subset is a segment of data from the data payload having a predetermined byte size.

74.   A method of analyzing network activity comprising:

obtaining a plurality of packets transiting a network;

performing a data reduction on at least a portion of each of the packets of said plurality of packets to form a plurality of data reduced packets;

analyzing said plurality of data reduced packets to detect a repetition of at least a portion of content among said plurality of data packets; and

analyzing said packets having repetitive content to determine if said packets are spreading.

75.   The method of claim 74, wherein analyzing for spreading comprises determining whether there is an increasing number of sources and destinations of packets having common content.

76.   The method of claim 74 wherein analyzing spreading comprises:
        monitoring a first content sent to a destination;
        monitoring a second content sent by said destination; and
        determining a whether there is a correlation between said first content and said second content.

77.   The method of claim 74 further comprising comparing the destinations of said plurality of packets with destinations having known vulnerabilities.

78.   The method of claim 77 further comprising analyzing the content of packets with destinations having known vulnerabilities.

79.   The method of claim 74 further comprising determining a signature of an attack based upon said analyzing of said plurality of data reduced packets and said analyzing spreading.

80.   The method of claim 74, wherein analyzing of said plurality of data reduced packets includes analyzing the content of the payloads in the plurality of packets.

81.   The method of claim 76 further comprising analyzing the content of said plurality of packets for the presence of a specified type of code.

82.   The method of claim 74 wherein said data reduction comprises carrying out a hash function on at least a portion of each of the packets.

83.   The method of claim 74 further comprising using at least first, second and third data reduction techniques on at least a portion of each of the packets, to obtain at least first, second and third results, and to count said first, second and third results, and to establish frequently occurring sections when all of said at least first second and third results have a frequency of occurrence greater than a specified amount.

84.   The method of claim 74 further comprising forming a plurality of portions from each of said plurality of packets, each portion comprising a specified subset of a packet.

85.   The method of claim 84, wherein a single portion comprises data from at least two packets.

86.   The method of claim 84, wherein analyzing further comprises analyzing a subset of the plurality of portions, the subset identified by a common characteristic in each of said portions.

87.   The method of claim 86, wherein the specified subset is a segment of data from the data payload having a predetermined byte size.

88.   An apparatus comprising:

a signature generator, having a connection to a network, to obtain a portion of data from the network, operating to carry out a data reduction on said data portion to reduce said data portion to a reduced data portion in a repeatable manner; and

a memory, storing said reduced data portions; and

wherein said signature generator also operates to detect common elements within said reduced data portion, said analyzing reviewing for common content indicative of a network attack.

89.   An apparatus as in claim 88, wherein said signature generator determining frequently occurring sections of message information within said reduced data portion.

90.   An apparatus as in claim 88, wherein said memory stores information indicative of at least one of a number of sources sending the common content, and/or destinations that are receiving the common content, and said signature generator determines whether said number is increasing.

91.   An apparatus as in claim 88, wherein said signature generator also analyzes for the presence of a specified type of code within said data portion.

44

92.   An apparatus as in claim 89, further comprising a module that carries out an additional test on said frequently occurring sections of message information after said signature generator determines frequently occurring sections of message information.

93.   An apparatus as in claim 92, wherein said additional test is a test to look for an increasing number of at least one of sources and destinations of said frequently occurring sections of message information.

94.   An apparatus as in claim 93, wherein said module is a module to look for code within the frequently occurring sections.

95.   An apparatus as in claim 88, wherein said data reduction by said signature generator includes carrying out a hash function on said portion of data.

96.   An apparatus as in claim 89, wherein said determining frequently occurring sections is done by using at least first, second and third data reduction techniques on each said portion, to obtain first, second and third results, and to count said first, second and third results, and to establish frequently occurring sections when all of said first second and third results have a frequency of occurrence greater than a specified amount.

97.   An apparatus as in claim 90, wherein said portion of data at includes a portion of the network payload.

98.   An apparatus as in claim 92, wherein said module maintains a first list of unassigned addresses in said memory; forms a second list of sources that have sent to addresses on said first list; and comparing a current source of a frequently occurring section to said second list.

99.   An apparatus as in claim 98, wherein said module data reduces information prior to storing in said list.

100.   An apparatus as in claim 98, wherein said module operates to first monitor a first content sent to a destination;
        second monitor a second content sent by said destination; and

determine a correlation between said first content and said second content as said additional test.

101.   An apparatus as in claim 100, wherein said first monitoring comprises monitoring multiple destinations, and said second monitoring comprises monitoring multiple destinations during a different time period than said first monitoring.

102.   An apparatus as in claim 101, wherein said first and second monitoring comprises data reducing information about said destinations, and storing at least one table in said memory about said data reduced information.

103.   An apparatus as in claim 97, further comprising a data portion module that obtains a specified portion of data from the network.

104.   An apparatus as in claim 98 wherein said portion of data further includes a portion of a network header.

105.   An apparatus as in claim 99, wherein said portion of a network header is a port number indicating a service requested by a network packet.

106.   An apparatus as in claim 98, wherein said portion of data comprises a first subset of a network packet including payload and header and wherein said data portion module further obtains a second subset of the same network packet for subsequent analysis.

107.   An apparatus as in claim 106, wherein said data portion module forms a plurality of portions from each network packet, each of said plurality of portions comprising a specified subset of the network packet.

108.   An apparatus as in claim 88, further comprising forming a plurality of portions from each network packet, each of said plurality of portions comprising a continuous portion of payload, and information indicative of aport number requested by a network packet.

109.   An apparatus as in claim 88, further comprising:

first and second hash generators, respectively forming first and second hash functions of said portions;

a first counter, with a plurality of stages, connected such that respective stages of said counter are incremented based on said first hash function;

a second counter, with a plurality of stages, and connected such that respective stages of said counter are incremented based on said first hash function.

110.   An apparatus as in claim 109, further comprising a module that checks said one of said stages of said first counter and said one of said stages of said second counter against a threshold, and identifies said portion as frequent content only when both said one of said stages of said first counter and said one of said stages of said second counter are both above said threshold.

111.   An apparatus as in claim 110, further comprising a frequent content buffer table storing specified frequent content.

112.   An apparatus as in claim 111, further comprising at least a third counter, and a third hash generator, taking a third hash of said portion, and incrementing a stage of said third counter based on said third hash, where said module identifies said portion as frequent content only when all of said stages of each of said first, second and third counters are each above said threshold.

113.   An apparatus as in claim 112, wherein said signature generator includes a sliding window portion that first obtains said portion by taking a first part of the message, and subsequently obtains said portion by taking a second part of the message.

114.   A apparatus as in claim 113, wherein at least one of said hash functions is an incremental hash function.

115.   An apparatus as in claim 98, wherein said signature generator operates to form a hash function of at least one of the source or destination address, to form a hash

47

value, first determine a unique number of said hash values, and second determine said one of source or destination numbers based on said first determine.

116.   An apparatus as in claim 115, wherein said count carried out by said signature generator further comprises scaling the hash value prior to said second determine.

117.   A apparatus as in claim 116, wherein said scaling comprises scaling by a first value during a first counting session, and scaling by a second value during a second measurement interval.

118.   An apparatus as in claim 88, wherein said memory stores a list of computers on the network, and stores an update level for each of said computers indicating which of said computers is susceptible to a specified kind of attack, and a module which monitors for said kind of attack only when the message is directed for a computer which is susceptible to said kind of attack.

119.   An apparatus of claim 118 where said module checks comprises checking for a message that attempts to exploit a known vulnerability to which a computer is vulnerable, as said specified attack.

120.   An apparatus as in claim 119, wherein said module checks for a field that is longer than a specified length.

121.   The method of claim 31, wherein the portion of data is obtained by storing a minimum length subsection comprising data from the two or more adjacent packets.

122.   The method of claim 32, wherein the common characteristic is defined by the reduced data portion being equal to a value in a set of predefined values.

123.   The method of claim 56, wherein the portion of data is obtained by storing a minimum length subsection comprising data from the two or more adjacent packets.

124.   The method of claim 46, wherein analyzing further comprises analyzing a subset of the data reduced portions, the subset identified by a common characteristic in each of the data reduced portions.

125.   The method of claim 124, wherein the common characteristic is defined by the reduced data portion being equal to a value in a set of predefined values.
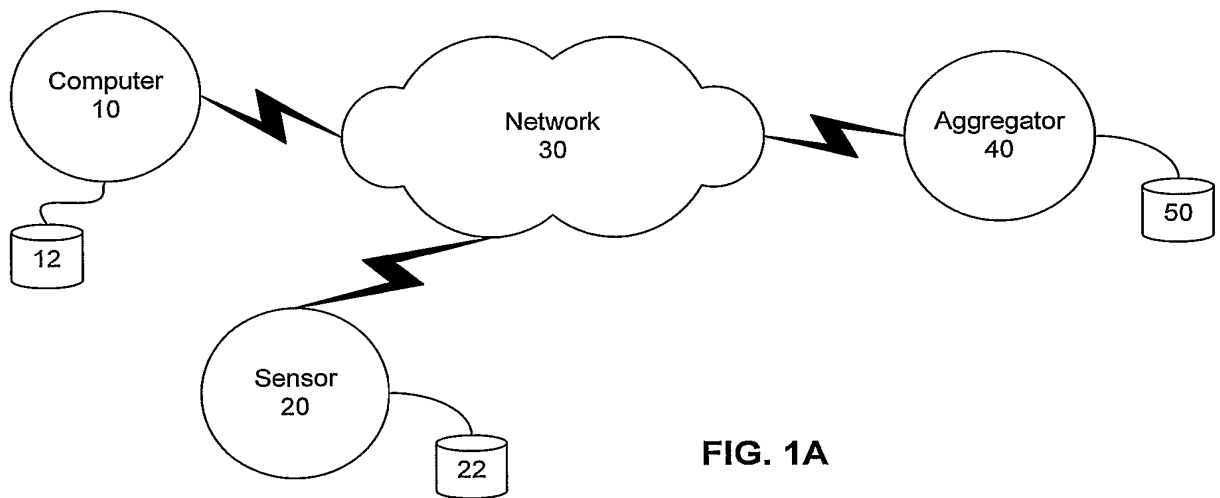
126.   The method of claim 71, wherein the single portion is obtained by storing a minimum length subsection comprising data from the two or more adjacent packets.

127.   The method of claim 70, wherein the common characteristic is defined by the reduced data portion being equal to a value in a set of predefined values.

128.   The method of claim 93, wherein the common characteristic is defined by the reduced data portion being equal to a value in a set of predefined values.
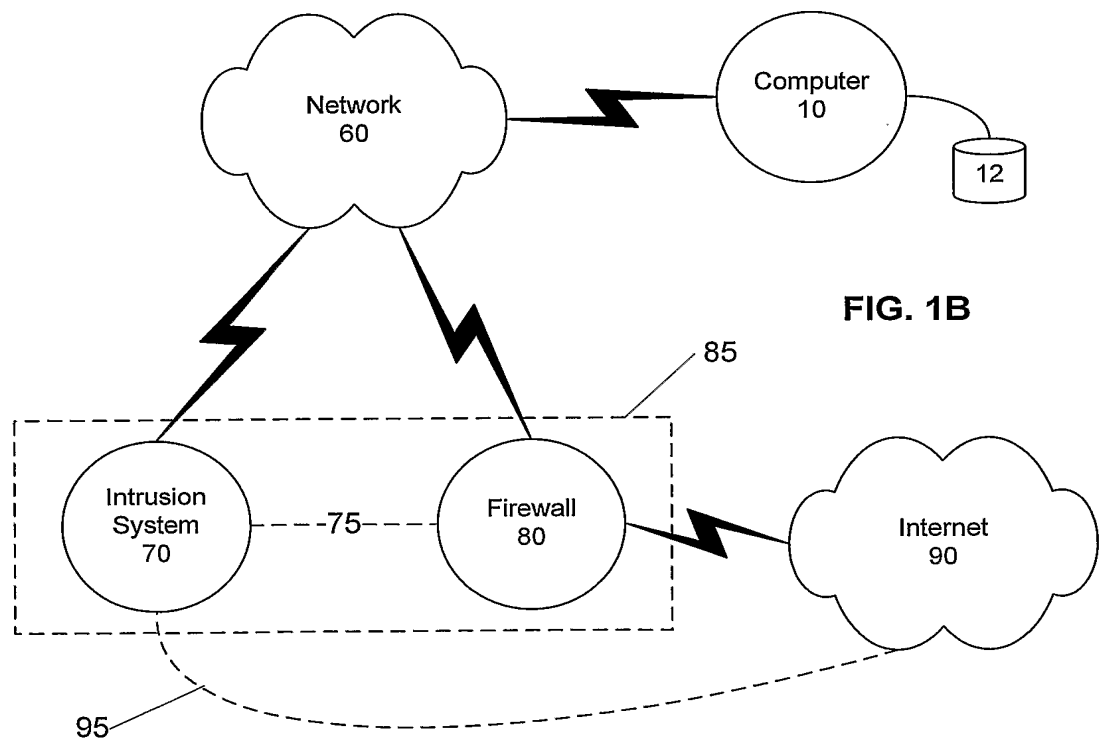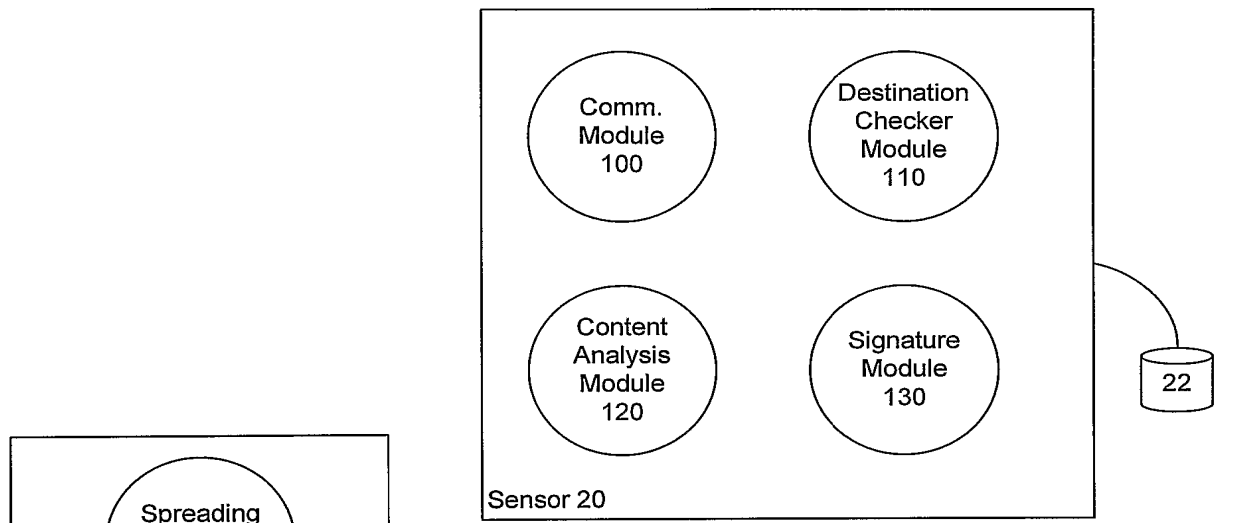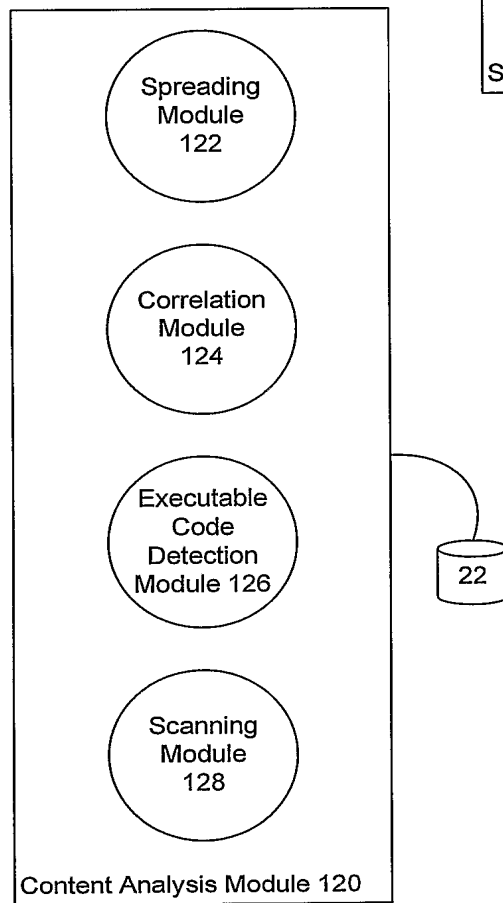
FIG. 1A

FIG. 1B
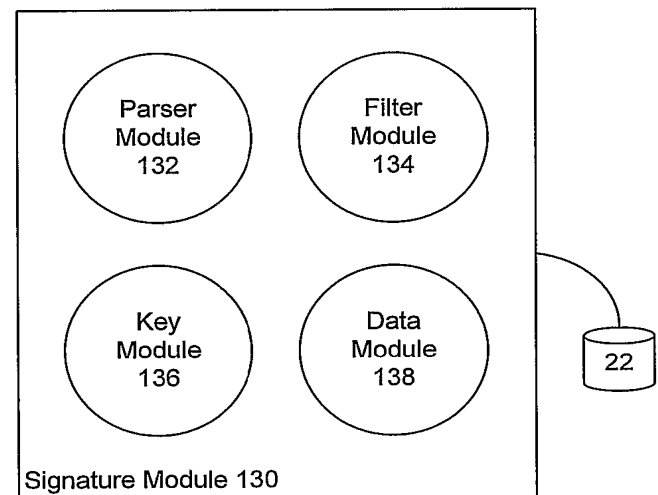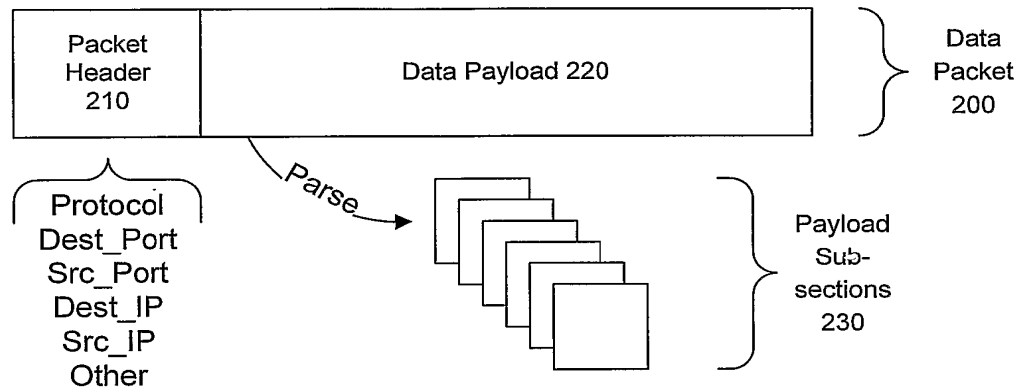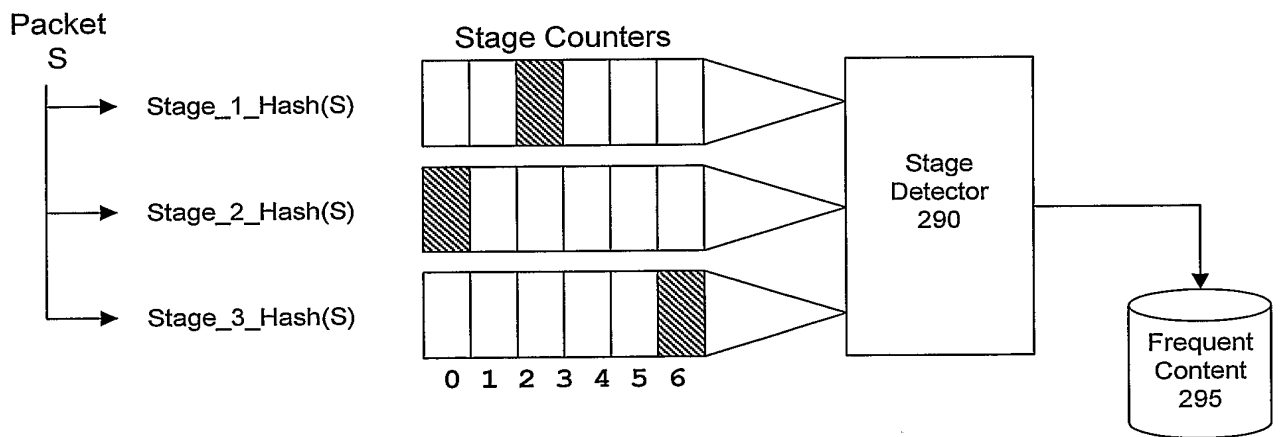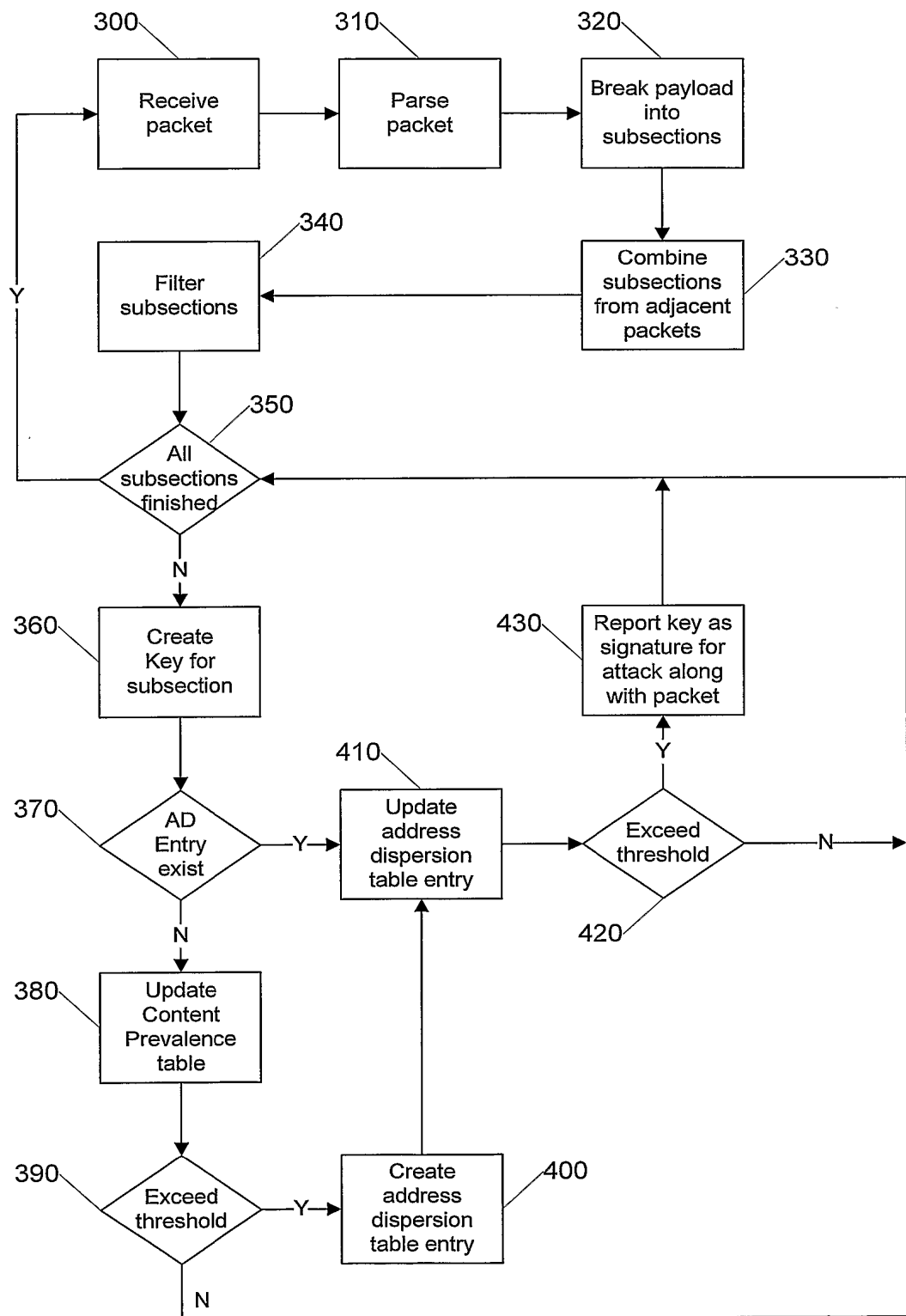
Page 2/4

Sensor 20

Comm.
Module
100

Destination
Checker
Module
110

Content
Analysis
Module
120

Signature
Module
130

22

FIG. 2A

Content Analysis Module 120

Spreading
Module
122

Correlation
Module
124

Executable
Code
Detection
Module 126

Scanning
Module
128

22

FIG. 2B

Signature Module 130

Parser
Module
132

Filter
Module
134

Key
Module
136

Data
Module
138

22

FIG. 2C

Page 3/4

FIG. 3

FIG. 4

FIG. 5

FIG. 6

Page 4/4

```
  ┌──────────┐      ┌──────────┐      ┌──────────┐
  │300       │      │310       │      │320       │
  │ Receive  │ ───▶ │  Parse   │ ───▶ │  Break   │
  │ packet   │      │  packet  │      │ payload  │
  │          │      │          │      │  into    │
  └──────────┘      └──────────┘      │subsections│
                                      └──────────┘
```

FIG. 7

A.	CLASSIFICATION OF SUBJECT MATTER
IPC(7)	: G06F 11/30, 12/14; H04L 9/00, 9/32
US CL	: 713/201

According to International Patent Classification (IPC) or to both national classification and IPC

B.	FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
    U.S. : 713/201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C.	DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	U.S. 2003/0115485 A1 (Milliken) 19 June 2003 (19.06.2003), figure 5 and paragraphs 25, 43-75	1-8, 10-16, 19, 25, 31-39, 43-46, 49-51, 53-59, 62, 66, 69-76, 79-82, 84-95, 97-104, 115, and 121-128 ------------ 9, 17-18, 20-24, 26-30, 40-42, 47-48, 52, 60-61, 63-65, 67, 77-78, 83, 96, 105-114, 116-119
Y	U.S. 2004/0064737 A1 (Milliken) 01 April 2004 (1.04.2004), paragraphs 18-49	9, 24, 26-27 52, 67, 83, 96, 106-107, 109-114, 116-117

☒ Further documents are listed in the continuation of Box C.	☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier application or patent published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 March 2005 (16.03.2005)	**11 APR 2005**

Name and mailing address of the ISA/US
    Mail Stop PCT, Attn: ISA/US
    Commissioner for Patents
    P.O. Box 1450
    Alexandria, Virginia 22313-1450
Facsimile No. (703) 305-3230
	Authorized officer
	Albert Decady
	
	Telephone No. (571) 272-3819

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	U.S. 2004/0054925 A1 (Etheridge et al.) 18 March 2004 (18.03.2004), paragraphs 16, 50-104	17-18, 20-23, 28-30, 40-42, 47-48, 60-61, 63-65, 77-78, 105, 108-114, 118-119
X,P	U.S. 2004/0073617 A1 (Milliken) 15 April 2004 (15.04.2004), full reference	1, 33, 43, 58, 74, 88
A	U.S. 2002/0107953 A1 (Ontiveros et al.) 08 August 2002 (08.08.2002)	1-128

C. (Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT